练习题2参考解答

一、

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
| **B** | **D** | **C** | **C** | **C** | **D** | **D** | **C** | **A** | **B** | **C** | **B** | **D** | **A** | **D** | **A** | **D** | **B** | **C** | **C** |

二、略，上机运行可知结果。

三、

1)

#include<iostream>

using namespace std;

class Vehicle // 定义汽车类 ，

{ protected:

int wheels; // 车轮数

float weight; // 重量

public:

Vehicle(int wheels,float weight);

int get\_wheels();

//getters & setters;

void show();

};

class Car:public Vehicle // 定义小车类

{

int passenger\_load; // 载人数

public:

Car(int wheels,float weight,int passengers=4);

//getters & setters;

void show();

};

class Truck:public Vehicle // 定义卡车类 ，

{

int passenger\_load; // 载人数

float payload; // 载重量

public:

Truck(int wheels,float weight,int passengers=2,float max\_load=24000.00);

//getters & setters;

void show();

};

Vehicle::Vehicle(int wheels,float weight)

{

Vehicle::wheels=wheels;

Vehicle::weight=weight;

}

void Vehicle::show()

{

cout << "车轮:" << wheels << "个" << endl;

cout << "重量:" << weight << "公斤" << endl;

}

Car::Car(int wheels, float weight, int passengers) :Vehicle (wheels, weight)

{

passenger\_load=passengers;

}

void Car::show()

{

cout <<" 车型:小车" << endl;

Vehicle::show();

cout << "载人:" << passenger\_load << "人" << endl;

cout << endl;

}

Truck:: Truck(int wheels, float weight,int passengers, float max\_load):Vehicle(wheels,weight)

{

passenger\_load=passengers;

payload=max\_load;

}

void Truck::show()

{

cout <<"车型:卡车" << endl;

Vehicle:: show ();

cout << "载人:" << passenger\_load << "人" << endl;

cout << "载重:" << payload << "公斤"<< endl;

cout << endl;

}

int main ()

{

Car car1(4,2000,5);

Truck tru1(10,8000,3,15000);

cout << "输出结果" << endl;

car1. show ();

tru1. show ();

return 0;

}

2)

#include<iostream>

using namespace std;

class Three\_d //类设计，

{ int A,B,C;

public:

Three\_d()

{ A=0;B=0;C=0;

}

Three\_d(int i,int j,int k)

{ A=i;B=j;C=k;

}

void get\_ABC(int &i,int &j,int &k)

{ i=A;j=B;k=C;

}

void show()

{ cout<<"A: "<<A<<",B: "<<B<<",C: "<<C<<"\n";

}

Three\_d operator++();

Three\_d operator++(int);

Three\_d operator--();

Three\_d operator--(int);

};

Three\_d Three\_d::operator++() //前置++

{

++A;

++B;

++C;

return \*this;

}

Three\_d Three\_d::operator++(int) //后置++

{ Three\_d t(\*this);

A++;

B++;

C++;

return t;

}

Three\_d Three\_d::operator--() //前置--

{ --A;

--B;

--C;

return \*this;

}

Three\_d Three\_d::operator--(int) //后置--

{ Three\_d t(\*this);

A--;

B--;

C--;

return t;

}

int main()

{ Three\_d op1,op(10,10,10);

op1=op++; op1.show();

op1=++op; op1.show();

op1=op--; op1.show();

op1=--op; op1.show();

return 0;

}

3)

#include<iostream>

#include<cstring>

using namespace std;

class Error{};

class String

{

char \*m\_ptr;

public:

String(char \*ptr){

m\_ptr = new char[**strlen(ptr)+1**];

strcpy(m\_ptr,ptr);

}

~String(){

delete m\_ptr;

}

String &operator+=(const String &str){

char \*s = new char[**strlen(m\_ptr)+ strlen(str.m\_ptr)+1**];

if(m\_ptr){

strcpy(s,m\_ptr);

delete[] m\_ptr;

}

strcat(s,str.m\_ptr);// appends str.m\_ptr to s

m\_ptr = s;

return **\*this**;

}

bool operator==(const String &str) const

{

return (::strcmp(m\_ptr,str.m\_ptr)==0);

}

char operator[] (int i) {

if(i < strlen(m\_ptr))

return m\_ptr[i];

throw Error();

}

**friend** **ostream&** operator<<(ostream &out,const String&);

};

**ostream&** operator<<(ostream &out,const String &str)

{

return out<<str.m\_ptr;

}

int main()

{

String s1("Hello "),s2("world!");

if(s1==s2)

cout<<"S1==S2\n";

s1+=s2;

cout<<s1<<endl;

int k=0;

**try** {

while(s1[k])

cout<<s1[k++];

}

**catch (Error e)** { //***也可以catch(Error& e)***

cout<<"\nIndex is out of range!"<<endl;

}

return 0;

}

4)

#include<iostream>

using namespace std;

class Shape{

public:

virtual float length( )=0;

};

float total(Shape \*s[],int n)

{

float sum=0.0;

for(int i=0;i<n;i++)

sum+=s[i]->length( );

return sum;

}

class Triangle : public Shape {

float e1, e2, e3;

public:

Triangle(float ei1, float ei2, float ei3):e1(ei1),e2(ei2),e3(ei3) {}

float length() { return e1+e2+e3; }

};

class Rectangle : public Shape {

float width, height;

public:

Rectangle(float w, float h):width(w), height(h) {}

float length() { return 2\*(width+height); }

};

int main(){

Shape \*s[2];

s[0]=new Triangle(3,4,5);

s[1]=new Rectangle(3,5);

cout<<s[0]->length()<<" "<<s[1]->length()<<" "<<total(s,2)<<endl;

return 0;

}